tei-workshop-instructions

Welcome to the Introduction to XML and Digital Scholarly Editing with the Text Encoding Initiative (TEI) workshop. Instructions and URLs needed to complete the workshop exercises are found below. These instructions are available online at https://paulbroyles.github.io/tei-workshop-instructions/.

Getting Started

To get started, you must go to the website for the TEI Web Editor app and authorize it by logging into your GitHub account. (If you have not already created an account at GitHub.com, you will need to do so before you begin.)

1. Visit https://tei-web-editor.herokuapp.com
2. Click the Login button in the upper right and enter your GitHub username and password.
3. Click the green button marked “Authorize scta-texts.”

Now you should see your GitHub username in the upper right of the screen. The app will remain authorized to access your GitHub account until you explicitly revoke access.

If you want to revoke access after the workshop is over, go to GitHub.com, click on your profile picture, and select Settings. From the menu on the left, select Applications. At the top of the page, open the tab labeled “Authorized OAuth Apps.” Press the button labeled “Revoke” next to “scta-text-ui-production.”

Exercise 1: Encoding Poetry

For our first activity, we will practice encoding poems. Working with poetry will allow us to see how texts are composed of components arranged hierarchically. We’ll see how a single document can contain multiple poems, which are made up of stanzas that are in turn made up of lines. We’ll also see how we can enrich a text with further tagging to aid analysis.

1. Click on the Open button at the top of the screen.
2. Scroll down to “Selected Repositories to Start From” and click on “Simple TEI Edition.”
3. in the box below the words “Create New Branch,” enter `gh-pages` and press “Create.”
4. Click on the `xml` folder, then on the `index.xml` file.

5. At the top of the screen, click on the giant plus sign (+) with the words “Add Item” below.
6. Copy this link: https://iiif.lib.harvard.edu/manifests/drs:43369310
7. Paste the link into the box next to the words “Add new object from URL” and click “Load.”
8. The top part of the screen now contains images from a book of Emily Dickinson’s poetry. To the left of this section, you will see the contents of the book. Choose a section, I through IV, and expand it. Then choose a page. You will add this poem to your edition. Zoom in as appropriate to make sure you can see the text.

So far, you have prepared your work environment by creating a new repository in your GitHub account, and you have loaded images of a primary source to serve as the basis of your edition. You are now ready to begin creating the edition.

1. Edit the TEI header as appropriate to give some sense of what your edition is and where it comes from.
2. Find the `<body>` element. Erase the `<p>` element; that represents a paragraph, which we don’t need here.
3. Open a new `<div>` tag. This represents an arbitrary division. Notice how Web Editor automatically creates the closing `</div>` tag for you.
4. For our purposes, we will treat each poem as a separate `<div>`. To make this clear, add the attribute `type="poem"` to the opening `<div>` tag.
5. Within the `<div>` element, create a new `<lg>` element. This means “line group.” We will use this element to contain a stanza (that is, a set of lines in a poem that are grouped together). To show this, add the attribute `type="stanza"` to the element.
6. Inside the `<lg>` element, create a new `<l>` element. This means “line,” and will hold a single line of poetry.
7. Transcribe the first line of poetry within the `<l>` element.
8. Go on to transcribe the remainder of the poem. Each line should be within its own `<l>` element; the lines that make up a stanza should be grouped together in a `<lg>` element.
9. If you have time, open a new `<div>` and transcribe the next poem in the book. (You may use the arrow buttons in the top area to turn the pages.)
10. Click the Save button at the top of the page. In the left column, enter a brief description of what you’ve done in the box marked “Message,” then scroll down and click `Save`.
11. At the top of the screen, click on the words `Current doc: index.xml`. Select `View on GitHub Pages`. A new tab will open, showing you that your poem is visible on the web.

Bonus Activity: Visit the TEI Guidelines, view the chapter on Verse, and look up how to encode rhyme. Add the appropriate markup.

Exercise 2: Encoding Manuscript Materials

For our second activity, we will practice encoding a letter—a common kind of manuscript you might want to encode in TEI. We will see how special elements allow us to mark up significant features of this kind of document, how TEI encoding allows us to record changes to the document like additions or deletions, how we can make editorial interventions like regularizing mistakes, and how we can record specific visual features of the document.

1. In a separate tab, visit https://github.com/paulbroyles/tei-workshop-letter and click the `Fork` button in the upper right.
2. Return to the TEI Web Editor tab and click on the Open button at the top of the screen.
3. Under `My Repositories`, click the link that ends `tei-workshop-letter`.
4. In the Branch column, click on `gh-pages`.
5. Click on the `xml` folder, then on the `index.xml` file.
6. At the top of the screen, click on the giant plus sign (+) with the words “Add Item” below.
7. Copy this link: https://d.lib.ncsu.edu/collections/catalog/ll000159/manifest.json
8. Paste the link into the box next to the words “Add new object from URL” and click “Load.”
9. Click on the first image in the new row that appears to load the letter.

Your workspace should now contain an image of a letter at the top. The editing window at the bottom left should include an element with elements as grandchildren. These elements will help us encode specific visual features of the letter.

This time, you will have to decide how to structure the file and what elements you should use. You will have to make choices about what features of the letter you wish to encode, and how to organize them. Should you separate the letter into parts? Do you want to record line breaks? Where the letter writer has made an obvious mistake, will you correct it? Feel free to discuss or ask questions as you go.

Elements relating to text structure

- `<opener>`: “groups together dateline, byline, salutation, and similar phrases appearing as a preliminary group at the start of a division, especially of a letter.”
- `<closer>`: “groups together salutations, datelines, and similar phrases appearing as a final group at the end of a division, especially of a letter.”

- `<dateline>`: “contains a brief description of the place, date, time, etc. of production of a letter, newspaper story, or other work, prefixed or suffixed to it as a kind of heading or trailer.”
- `<salute>`: “(salutation) contains a salutation or greeting prefixed to a foreword, dedicatory epistle, or other division of a text, or the salutation in the closing of a letter, preface, etc.”
- `<signed>`: “(signature) contains the closing salutation, etc., appended to a foreword, dedicatory epistle, or other division of a text.”
- `<p>`: paragraph
- `<lb/>`: (empty element) line beginning. Used to indicate that a new line has started, where that is not already made clear by other tagging.

Elements relating to changes to the text.

- `<add>`: “(addition) contains letters, words, or phrases inserted in the source text by an author, scribe, or a previous annotator or corrector.”
- `<del>`: “(deletion) contains a letter, word, or passage deleted, marked as deleted, or otherwise indicated as superfluous or spurious in the copy text by an author, scribe, or a previous annotator or corrector.”
- `<choice>`: contains two or more elements that represent alternate encodings of the same point in the text.
    - `<sic>` / `<corr>`: Pair of elements used when an editor wishes to correct an error in the text. Within a `<choice>` element, the `<sic>` element contains a string of text exactly as it appears in the source document;
      the `<corr>` element contains an editorially corrected version of the same text.
- `<hi>`: contains text that is highlighted or marked out in some way by its visual appearance. The `<hi>` element has no real meaning without an attribute,
  like `rendition`, that encodes its appearance.

The `rendition` attribute

The TEI offers various mechanisms for encoding the visual appearance of text. One of these mechanisms is the `rendition` attribute, which indicates the visual appearance of the text contained within an element. The appearances associated with `rendition` are set in the header, where a set of elements can describe the appearance of text in a language like CSS. Each `rendition` element can be given an `xml:id` attribute; any element in the text can then be given that appearance by adding the `rendition` attribute, with a value equal to the `xml:id` of one of the `rendition` elements in the header, prefaced by the `#` sign. For this example exercise, I have supplied the following `rendition` values:

- `right`: This element is right-aligned on the page.
- `del-x`: The text in this element has been marked out with an X.

- `del-dbl`: The text in this element has been marked out by being crossed out with a double line.
- `ord`: Text written superscript and underlined, for example the “th” in the ordinal number “19th.”
- `sup`: Text written in elevated or above the line, for example added text placed above the line.

Think about the choices you made in encoding the material. What features did you choose to indicate with markup? What did you ignore? What important characteristics of the text could not be represented by the elements we have learned? What other approaches might you have taken?

Bonus Activity: Visit the TEI Guidelines, view the chapter on “Names, Dates, People, and Places,” and consider how you might use that set of elements to add metadata to your edition of the letter. How would different choices affect the way you can use your edition?

Resources Mentioned in the Workshop

Learning TEI

- TEI by Example (http://teibyexample.org/)
- TEI-L Listserv (https://listserv.brown.edu/archives/cgi-bin/wa?A0=TEI-L). Best for specific questions after you’ve consulted other resources.
- The TEI Guidelines (http://www.tei-c.org/release/doc/tei-p5-doc/en/html/)

Software

- oXygen (https://www.oxygenxml.com/). Commercial XML editor.

Tools for publishing TEI

- CETEIcean (http://teic.github.io/CETEIcean/). Simple JavaScript library to display TEI editions directly in the browser.
- TEI Boilerplate (http://dcl.ils.indiana.edu/teibp/index.html). Uses XSLT to display TEI files directly in the browser. Older technology; I recommend CETEIcean instead.
- TEI Publisher (https://teipublisher.com/index.html). Tool to create webapps for TEI editions.

- EVT/Edition Visualization Technology (https://visualizationtechnology.wordpress.com/). Tool to create TEI diplomatic editions and display them along with facsimile images.